CSC2535 2013
ASSIGNMENT 2

Geoffrey Hinton and Navdeep Jaitly

February 15, 2013

Assignments are due by 3.00pm on wednesday Feb 27. Printed copies are required. This assignment is worth 20% of the final grade and is marked out of 20 points.

Late assignments will have 20% subtracted from the total out of which they are graded for each day or part of a day that they are late. They will be more than one day late if they are not slipped under my office door (LP 290G) before 3.10pm the next day.

Problem 1 (5 points)

The CD1 training procedure for a Restricted Boltzmann Machine that uses stochastic binary units has some similarities to the training procedure for an autoencoder that uses deterministic logistic units and a cross entropy error for each visible unit.

Show that the when the reconstructions are fairly accurate, the CD1 learning procedure is a good approximation to following the derivative of the cross entropy reconstruction error.

Problem 2 (3 points)

A feed-forward neural network that has N linear output units can be trained to assign probabilities to classes in the following way:

Before training begins, each class, c, is assigned a fixed "code vector" Y_c composed of N real-valued components. This code vector never changes.

The probability that a net with output vector V assigns to class c is given by:

$$p(class = c|V) = \frac{\exp(-||Y_c - V||^2)}{\sum_i \exp(-||Y_i - V||^2)} \tag{1}$$

Show that this is equivalent to using an additional hidden layer followed by a softmax output layer.

Problem 3 (2 points)

At a temperature of 1, the free energy, F_Ω, of a set of alternative configurations, Ω, is defined by:

$$\exp(-F_\Omega) = \sum_{i\in\Omega} \exp(-E_i) \tag{2}$$

where E_i is the energy of configuration i.

Show that F_Ω is equal to the minimum over all possible probability distributions of the variational free energy which is defined by:

$$V_\Omega = \sum_{i\in\Omega} p_i E_i + \sum_{i\in\Omega} p_i \log p_i \tag{3}$$

where p_i is the probability of configuration i.

Problem 4: Alternative ways of learning RBM's (10 points)

Empirically evaluate the following three methods of learning a restricted Boltzmann machine:

a) **CD1**. This starts at the data and uses three half steps of alternating Gibbs to update the hidden units, then the visible units, then the hidden units again. So it uses one full step of alternating Gibbs to produce the negative data that are used for approximating the gradient of the partition function (see lecture notes).

b) **CD10**. This uses 10 full steps of alternating Gibbs to produce the negative data.

c) **Persistent CD**. After each weight update this runs one full step of alternating Gibbs starting with the current negative data to produce new negative data. Before learning starts the negative data is initialized at random data.When learning involves multiple mini-batches of data, the negative data is started at the negative data that was used for the previous mini-batch, so only one mini-batch of negative data is used. To evaluate these three methods, you should find a dataset of binary data that is highly structured (so there is a significant amount of structure for the RBM to learn) and is not too high-dimensional (so you dont need too many parameters). For example, an RBM with 100 hidden units trained on 1000 binarized 1616 images of handwritten twos would do, but you should attempt to find some other dataset. The hope is that different students will get results on different datasets. There is no need to try more than one dataset unless you decide that the first one you tried is inappropriate. Some simple code for training an RBM with CD1 can be found at
`http://www.cs.toronto.edu/~hinton/code/rbm.m`

The learning rate and weightcost in this code may need to be changed for your dataset. To compute the log probability that a learned RBM assigns to your test data, you will need to compute the negative free energy (the goodness) that the model assigns to each test vector and subtract the log of the partition function for that RBM. Matlab code for computing the log of the partition function is provided at: `http://www.cs.toronto.edu/~hinton/code/partition/` The folder is not readable so you should download the files individually. The files are:

```
free_energy.m -- computes free energy for use in RBM_partition.m
logdiff.m and logsum.m -- used for computing log(Z+-3std)
RBM_partition.m -- is the main engine for calculating Z
demo.m -- loads some data and an RBM model trained with CD25,
runs RBM_partition.m and returns log(Z),log(Z+3std),log(Z-3std).
Occasionally you may get logZZ_down=0.000000.
If this happens just try again.
```

Computing the partition function accurately can take a long time, so you may only be able to do it a few times while learning an RBM. You can get a noisy estimate more quickly by using fewer intermediate distributions, but 5000 is already a very small number.

We do not expect you to do a polished piece of research. It is sufficient to find a single sensible dataset, train the three methods using a sensible number of hidden units, and compare the log probabilities they assign to your test data at a few sensibly chosen points during the learning. Question 3 should take you between 8 and 16 hours. Your answer should consist of up to one page describing your dataset, about one page describing your experiments, and one or two graphs showing how the log probabilities that the three methods assigned to the training and test data varied as they learned.